Homework 02 **Himen Hitesh Sidhpura**
COMP6651 **40091993**
Prof. Tiberiu Popa Due Date: 12/Apr/2019

1. Exercise 22.2-8

   Let x and y be the two end nodes of tree.
   To compute diameter of tree, we will use BFS algorithm with queue to find a longest path between two nodes.
   Step 1 : Run BFS algorithm from root node of tree to farthest nodes x.
   Step 2 : Now consider x as root and run BFS again to farthest node y and Consider diameter equal to 1 at root level.
   Step 3 : At each level of tree, increment diameter by 1.
   Step 4 : Once we reached y, we will get longest path in tree which is diameter.
   Since complexity of BFS is $O(n)$. Overall Complexity to computer diameter will $O(n)$, where n is number of nodes in tree.

   Please Refer BFS Algorithm from Reference 1 Link

2. Exercise 22.3-13

   Step 1 : Create a list of vertex in graph.
   Step 2 : Perform DFS for every vertex as a source in graph and mark them as visited in list.
   Step 3 : While performing DFS, If visited vertex encountered again. We can conclude that graph is not singly connected. If there is no encounter of visited vertex then we will perform Step 4.
   Step 4 : Repeat Step 2 for all vertex which are unvisited in list.
   Step 5 : It is singly connected.

   Hence, Overall Complexity of this algorithm is $O(V^2)$, here V is vertex.

   Please Refer DFS Algorithm from Reference 10 Link

3. Exercise 23.1-11

   Step 1 : Add a newly decrease edge in tree. Addition of this edge will make the cycle. This will make tree once again graph G_1.
   Step 2 : On Graph G_1 apply Kruskal Algorithm.
   Step 3 :In this algorithm, first all edge will sorted.
   Step 4 : from this sorted edges, we will consider edges which has less weight and doesn't make cycle. If it make cycle, discard it.
   Step 5 :Repeat Step 4 for (V - 1) times. Here, V is total number of vertex in Graph G_1.
   Hence, Overall Complexity will be $O(ElogE)$ where E is number of edge in Graph G_1

   Please Refer Kruskal's Algorithm from Reference 5 Link

4. Exercise 26.3-4

   Let consider that there exist matching in G. Then each vertex of A will matched with R for any

subset $A \subseteq L$. There will be no two vertices matched with R, since there exist a matching. Therefore, there exist only $|A|$ vertices in A.

Now consider for all $A \subseteq L$, $|A| \leq N(A)$. Run Ford-Fulkerson on flow network. When augmenting path is found, flow in incremented by 1. This should happeFor corollary 26.11, n for $|L|$ times. Suppose, it happen less then there will no path. Then there exist fewer L edge from L to R having flow 1.

The flow is increasing each time by f_p by corollary 26.3. By corollary 26.11, the cardinality of maximum Matching in G is equal to value of maximum flow f. Since, $|L| = |R|$, maximum matching is perfect matching.

For corollary 26.3, Please Refer TextBook Page number 720
For corollary 26.11, Please Refer TextBook Page number 735

Corollary 26.3 : Let G = (V,E) be a flow network, let f be a flow in G, and let p be an augmenting path in G_f.
Corollary 26.11 : The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in its corresponding flow network G'.

References

[1] https://www.geeksforgeeks.org/diameter-n-ary-tree-using-bfs/

[2] https://www.geeksforgeeks.org/connectivity-in-a-directed-graph/

[3] https://www.geeksforgeeks.org/level-order-tree-traversal/

[4] https://www.geeksforgeeks.org/maximum-bipartite-matching/

[5] https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

[6] https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/

[7] https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols

[8] https://en.wikipedia.org/wiki/Matching_(graph_theory)

[9] https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[10] https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/